Simulation of the Impact of Packet Errors on the Kademlia Peer-to-Peer Routing

T. Ginzler
M. Amanowicz

tobias.ginzler@fkie.fraunhofer.de, marek.amanowicz@wat.edu.pl

ABSTRACT

Tactical radio communication is crucial for command and control. The progress in C2 software demands higher and higher transmission rates, more employed radios and higher robustness. Interconnectivity between various participants of a tactical network is achieved by increased usage of the Internet Protocol. Structured overlay networks, so called peer-to-peer networks can help to increase the reliability of IP-connections. The principle of peer-to-peer networks is to use a broad number of peers to transmit and retrieve messages instead of a single connection to a server. The redundancy increases its suitability for disadvantaged networks. At the time peer-to-peer networks became widespread, dial-up users with very little available bandwidth were common. This scenario is comparable to tactical scenarios where a limited number of base communication facilities and a larger number of mobile tactical radios exist. The error rate of dial-up lines and radios are both comparatively high. Tactical communication is even more supposed to be prone to errors and presumably suffers from higher packet error rates. Not all tactical user terminals are equal in capabilities such as transmission quality or battery power. It would be desirable if a peer-to-peer system respects these facts or could adapt to the situation to increase the responsiveness of such a system.

This paper will describe changes made to a widespread peer-to-peer system. Measurements in a simulated environment show that the speed of the overlay network can be increased by a slight modification of the routing. We incorporate cross-layer information about the error rate and delay of a connection or remaining battery power of a node into the overlay network to improve the responsiveness or traffic distribution of an existing peer-to-peer system.

1.0 THE KADEMLIA PEER-TO-PEER SYSTEM

The Kademlia peer-to-peer routing algorithm was chosen as a basis for further improvement, as it is prepared better for tactical environments than many of its competitors [2]. All peer-to-peer systems can be seen as a combination of a routing layer, a distributed data structure and a graphical user interface (optional). The user interface layer is not in the scope of the paper. The routing layer constantly generates a small amount of maintenance traffic. This traffic is considered in the paper as well as user traffic generated by the usage of the peer-to-peer functionality. The routing layer is responsible for finding nodes inside the peer-to-peer network. The Kademlia routing algorithm is complete, which means a search conducted by the routing either finds the node searched for or - if the node is not in the network - the search terminates with an appropriate message. The routing is based on an algorithm working on a key space with a metric to define distances. Overlay networks which share this property are called structured peer-to-peer networks. As a unique feature the Kademlia routing algorithm uses incoming routing requests to learn about its surrounding nodes. This makes the algorithm more bandwidth efficient than other structured peer-to-peer overlay protocols.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2010	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Simulation of the Impact of Packet Errors on the Kademlia Peer-to-Peer Routing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **NATO**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADA568727. Military Communications and Networks (Communications et reseaux militaires). RTO-MP-IST-092

14. ABSTRACT
Tactical radio communication is crucial for command and control. The progress in C2 software demands higher and higher transmission rates, more employed radios and higher robustness. Interconnectivity between various participants of a tactical network is achieved by increased usage of the Internet Protocol. Structured overlay networks, so called peer-to-peer networks can help to increase the reliability of IP-connections. The principle of peer-to-peer networks is to use a broad number of peers to transmit and retrieve messages instead of a single connection to a server. The redundancy increases its suitability for disadvantaged networks. At the time peer-to-peer networks became widespread, dial-up users with very little available bandwidth were common. This scenario is comparable to tactical scenarios where a limited number of base communication facilities and a larger number of mobile tactical radios exist. The error rate of dial-up lines and radios are both comparatively high. Tactical communication is even more supposed to be prone to errors and presumably suffers from higher packet error rates. Not all tactical user terminals are equal in capabilities such as transmission quality or battery power. It would be desirable if a peer-to-peer system respects these facts or could adapt to the situation to increase the responsiveness of such a system.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**12**	

The Kademlia peer-to-peer system was first described by Maymounkov and Mazieres [7]. It belongs to the class of prefix-based structured peer-to-peer overlay networks. In particular it is different from numerical distance systems such as Chord [3], unstructured overlay systems as Gnutella and super peer networks such as Skype [4].

1.1 Routing

A participant of the overlay network is called a *node* inside the overlay. The routing component in a peer-to-peer system is responsible for determining the location of a node in the overlay network. Every node has a unique identifier of fixed length. In Kademlia the length of the identifier is 160 bit. Nodes choose their identifier randomly and uniformly distributed at startup so the identifiers are unique with high probability, even with millions of participants.

Every structured peer-to-peer network has at least one metric to calculate a distance between two nodes. Kademlia uses the XOR metric. The distance between two nodes is calculated by bitwise applying the XOR operator to the two identifiers of the nodes. The resulting bit pattern is then treated as a number. As the XOR operation is commutative, the metric is symmetric. That means *A* has the same distance to *B* as *B* has to *A*. Also, A XOR A is 0.

Every node in Kademlia has a routing table. The table consists of *buckets* which hold a fixed number (*k*) of references to reach other nodes. Due to memory considerations the size of the table has to be limited. In Kademlia the memory requirement for the routing table is $O(k \cdot b)$, with b as the number of bits of an identifier. The routing table is organized as a tree as shown in Figure 1. The buckets are shown in the figure as rectangles with references to the identifiers in it. A node carries a tag which defines which identifiers are contained in its subtree. In the figure, *b* and *k* are assumed to be 4 for simplification. A tag of 1*xxx* means that the highest value bit is 1 for the whole subtree and the other bits are unknown. The right subtree of the root carries this tag. The local node identifier is assumed to be 0000 in the depicted tree. The tree structure can be seen as a general structure, if the tags and bucket entries are assumed to be XORed by the local node's identifier.

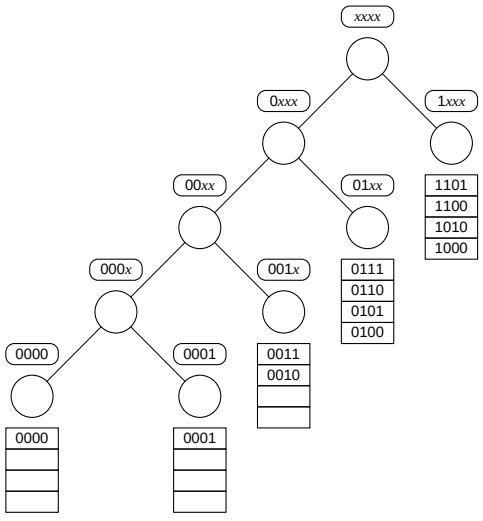

Figure 1: Kademlia routing table with b=4

The two subtrees below the root separate the identifier space in two halves: One subtree contains references to nodes closer to the local node than half of the maximum distance (the left side) and the other one contains references to nodes further away (the right side). The rightmost bucket holds *k* references to nodes which differ in their most significant bit from the identifier of the local node. These nodes the

references point have a maximum distance of half the distance to the destination nodes as the local node has. The left subtree is constructed recursively with increasing matching prefix length. This enables the node to store more references to closer nodes than to node which are far away. The leftmost bucket contains only the reference to the local node, its sibling bucket may hold exactly one node which differs only in the least significant bit. It is less and less likely that the buckets to the left are filled the farther left they are. That is due to the equal distribution of the identifiers.

When a node is called by another node to look up an identifier, the node starts looking for the nearest reference to that node it has in its table (Figure 2). To do so, the tree is traversed beginning from the root and that subtree which matches the most significant bit is chosen. If the traversal continues in the left branch the next bit is considered until a bucket is found. If the chosen bucket is empty, the bucket to the right is chosen. This approach is equivalent to XORing the destination identifier with the own node identifier and choosing the bucket corresponding to the leading zeros of the resulting bit pattern. The node returns the references of the bucket to the caller.

The caller collects all incoming contact references in a separate list named *L*. It then reissues lookup requests to the nodes closest to the destination. The closest nodes are most likely found in *L* but may also come from the routing table. The algorithm terminates if *r* closest references fail to return references closer than already known to the caller.

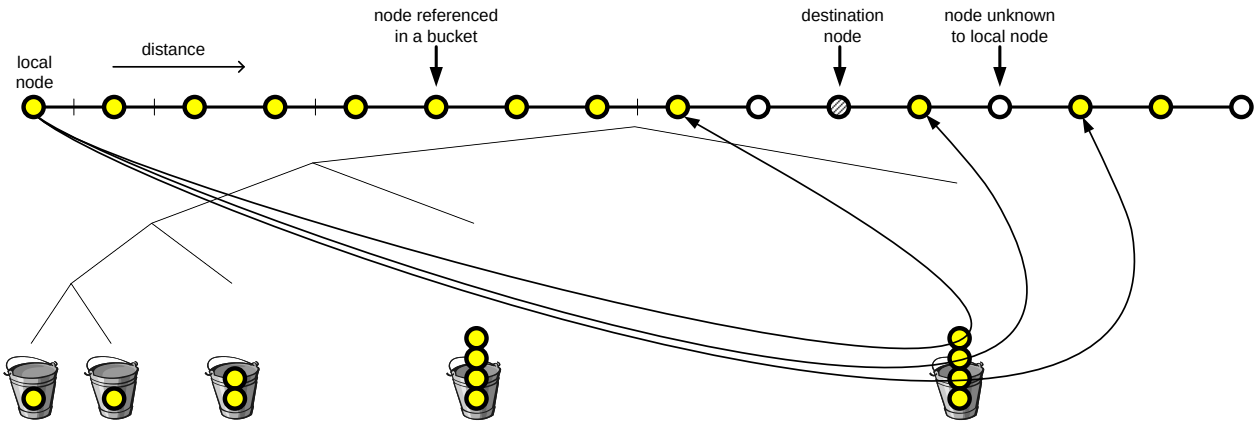

Figure 2: The original algorithm always chooses the known nodes closest to the destination, k=4

This lookup scheme is guaranteed to terminate as it is monotonic and the distance to the destination node is at least cut in half in every routing hop. The number of hops to the destination is bound by $O(\log n)$ with n being the number of nodes in the network. Uniformly distributed identifiers and reasonably filled tables are assumed. The initial filling of a node's routing table is achieved by contacting a node in the overlay and issuing a search for the own identifier.

The wording "close" and "distance" are in terms of the overlay metric. A node close to another from an overlay point of view may be very far away in terms of a lower layer metric. A hop in the overlay may include many hops in lower layers. From this fact arises the issue that lookups may take very long, as there is a discrepancy between the two metrics. The changes proposed in this paper address this issue.

1.2 Distributed Hash Table

The distributed hash table (DHT) provides two API calls:

1. STORE a tuple of (key, value)

Before information may be stored it is assigned unique key. The key may be derived from a textual description, a file name or is a structured number such as an object identifier. Value means any kind of data. In existing peer-to-peer systems a reference to the storage location of the - possibly large - amount of data is stored.

2. GET a value associated with a key

This retrieves the value associated with a key which was stored previously.

To store a value, the DHT uses the overlay routing. The overlay network is used to find a node inside the overlay network with an identifier close to the key. There is an implicit mapping between the key of the data and the location of nodes in the network. Data is always stored at the nodes with identifiers closest to the key of the data. A redundancy factor defines how many nodes are chosen for storing a value to prevent the loss of data if nodes leave the network. In the Kademlia simulation we assume the data is stored at the *k* closest nodes.

2.0 MODEL OF A TACTICAL ENVIRONMENT

We made some assumptions on what we expect a tactical network to contain. We expect 256 nodes to take part in a peacekeeping mission or disaster recovery operation. The nodes are interconnected with an IP connection in a fully meshed manner with a delay based on the distance of the nodes. The delay increases linearly with the distance. The maximum channel delay is 7 s. This roughly models the effect of a Layer 2 multi-hop ad-hoc routing protocol. Every node features a bit error rate of either 0 with a probability of 0.3 or a bit error rate of 10^{-3} which remains constant over the simulation time. This models heterogeneous node connectivity. Some nodes are connected by a wired connection and some feature an unreliable radio connection. A screenshot of the simulation model is shown in Figure 3. The simulated nodes remain static and do not move as a simplification.

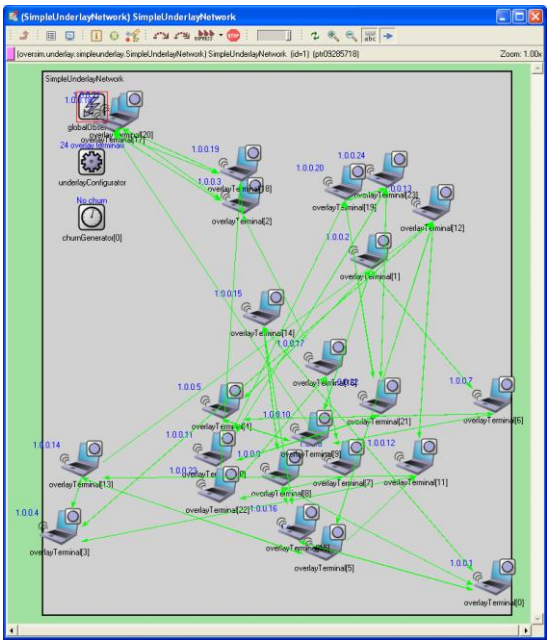

Figure 3: Screenshot of the simulation - the arrows indicate an overlay connection

The nodes report information about their internal state to the overlay routing. This feature could be offered e.g. by existing IEEE 802.11 WLAN hardware utilizing a mechanism called expected transmission count (ETX). The interface to the overlay is added to the OverSim framework [13] which was utilized for these measurements. We measure the effect of incorporating cross-layer information into a peer-to-peer system. The cross-layer information about packet error rate or any arbitrary lower layer metric is given to the overlay in form of a dimension-less *preference value*.

The preference value reflects the fact that some nodes should be preferred by the overlay as they have or lack special capabilities or limitations. A connection to a node with a low packet error rate is more likely to handle an overlay connection correctly than a connection with a higher bit error rate. The preference originates from the layers below the overlay, e.g. the operation system or from the hardware. The preference of a node is published to other nodes, which may combine these values with locally observed values. How the preference values may be transferred between nodes is out of scope. Attaching a preference value to node contact information may be an appropriate method to do so. A preference is derived from the tuple (*s*,*d*) of the local source node and a destination node. The preference may also be calculated from the preference of the remote node only. In our measurement we assume the reported preference values to be always correct.

3.0 MODIFICATION OF THE ROUTING

We incorporate cross-layer information about the error rate of a node into routing decisions of the overlay network and we show that this change makes the overlay routing more successful under difficult conditions. Several changes to the original Kademlia algorithm are proposed, many of them apply to other structured overlay networks as well. An overview is given in [6]. Our approach is not to modify any existing routing parameters but to use a different method of choosing contacts. Thought the method is also applicable to other peer-to-peer systems, the scope of this paper is limited to Kademlia. This additional cross-layer information is called preference value or simply the preference of a node.

The routing described in Section 2.1 is changed in the way the caller selects nodes to contact as it receives contact references from other nodes. The original Kademlia chooses contacts which are closest to the identifier to look up. In our setup we have a preference value available about every contact. So the order of the entries of the list *L* is changed to put contacts with a higher preference value closer to the head of the list.

We introduce a weight factor *w*, which determines the influence of cross-layer information. If the weight factor is 0 the routing behaves like an unchanged Kademlia routing as the preference has no influence on *L*. If the weight factor is 1, the decision is purely based on the preference metric and *L* is sorted according to the preference values. Intermediate values of the weight affect the order in a continuous manner. The new sorting order is defined by:

$$m_d = weight \frac{pref_{(s,d)} \cdot length(L)}{\max\_ pref} + (1 - weight)\, pos_d,$$

where *s* denotes the local source node making the routing decision and *d* a remote destination node. The original position of *d* the in the list *L* is pos_d. The list *L* is then reordered in descending order according to m_d.

The change to the routing is fully compatible with existing clients in the network. Especially the correctness proof of the Kademlia routing algorithm is untouched, as still in every routing step the distance to the destination is cut at least by half. That comes from the fact that the bucket selection remains unchanged. The reordering of *L* may be useful in other prefix based routing peer-to-peer networks as well. The advantage of this scheme over e.g. Pastry's inclusion of cross-layer information is that the metric remains the same during the routing process. Pastry [15] switches to a proximity based metric when approaching a node closely. This complicates the implementation.

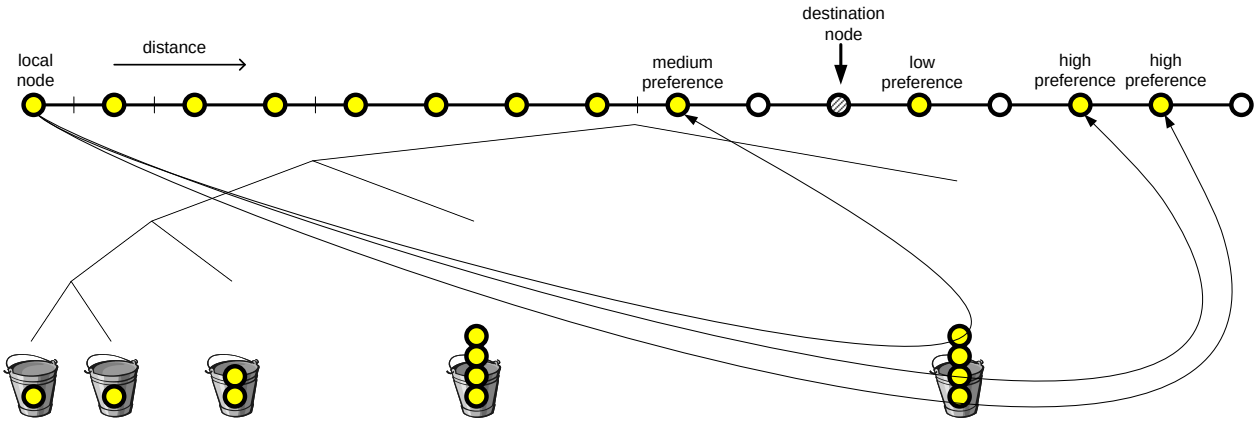

Figure 4: Another set of nodes is chosen due to higher preference

We test three methods to generate a preference value. The first method is to take the channel delay between the local node and the reference *i* of list *L* as preference value. After a normalization step the delay is used as preference. We call this modification *modification 1*. The second method is to take the bit error rate between the local node and the remote node $BER_{(s,d)}$ as a preference value, it is called *modification 2*. *Modification 3* takes a value defined by the remote node only into consideration. The node may set a high value to attract routing traffic or a low value to avoid it.

4.0 MEASUREMENT SETUP

Simulator

We use the OMNeT++ simulator version 4.1 [14] with INET framework and the OverSim-20090908 extension [13]. We added an extension to incorporate cross-layer information into the routing decision of Kademlia to the OverSim framework and added capabilities to measure the effect of our changes.

The simulation time of a single run was 4 hours and 40 minutes. The initial 40 minutes of simulation time are used for stabilization of the network and no measurements takes place. A measurement with the same set of parameters is repeated 50 times with different random number seeds. The random seed influences the position of the nodes and thereby the delay distribution in the simulation. To speed up simulation, we used 14 cores on different machines to conduct simulations in parallel.

Metrics

The effect of the changed routing is measured by the DHT GET success rate of routing requests. Every node in the simulation issues a STORE or GET request every 100 seconds. The key of the STORE request is drawn from a uniform distribution. The value has a payload size of 100 byte and contains random data. A GET command asks for a randomly chosen key previously stored to the DHT. It then compares the resulting value with the expected value. If an answer is received and the value matches, the GET request is counted as successful. A successful GET requests for a key is a request which delivers the value previously stored under that key. The GET success ratio is the number of successful requests divided by the number of all GET request. A success rate below 1 is the result of network errors such as timeouts or packet loss or the effect of members leaving the system. In our simulation we do not consider the leave of nodes for the moment. The GET success rate is a metric for the reliability of the system. In existing implementations values need to be republished periodically otherwise they are dropped by the storing nodes. The lifetime and the republish interval of the values were chosen to be infinite to exclude side effects from the measurement.

We measure the time it takes to get the response of a successful GET request. The GET delay is a metric for the speed of the system. We focus on lowering the delay in the described environment.

The amount of data a node sends is recorded to measure the network load introduced by the overlay. The traffic is partly maintenance traffic which includes responses to routing requests and routing requests by the local node and partly traffic is generated by storing and retrieving values. No other traffic was present in the simulated network. The amount of traffic generated by a node is a metric of the load caused by the upkeep of the overlay network. As network resources are scarce, the generated amount of traffic has to be kept low.

5.0 RESULTS

In this section we present the results of the three modifications and describe the effects of the changes to the overlay network. The DHT GET success ratio is shown in Figure 1. The success ratio is 0.6, meaning that nearly half of the accesses to the DHT may fail. This is a consequence of the difficult communication environment described in section 3. For low reliability traffic such as weather information a lower success rate may not matter. While other communication systems may fail because of the described conditions, a peer-to-peer system still makes sense. The success ratio is unaffected or slightly increased by modification 1 and modification 2. The success rate is nearly unaffected because the storage destination of a value is only defined by its key and is not affected by the modification. So if the nearest nodes close to the key suffer from heavy packet loss, the data is supposed to be lost regardless of the modification. The modifications may only affect the hops towards the final storing node. The measurements show that the modifications also do not have a negative effect on the success rate of the system.

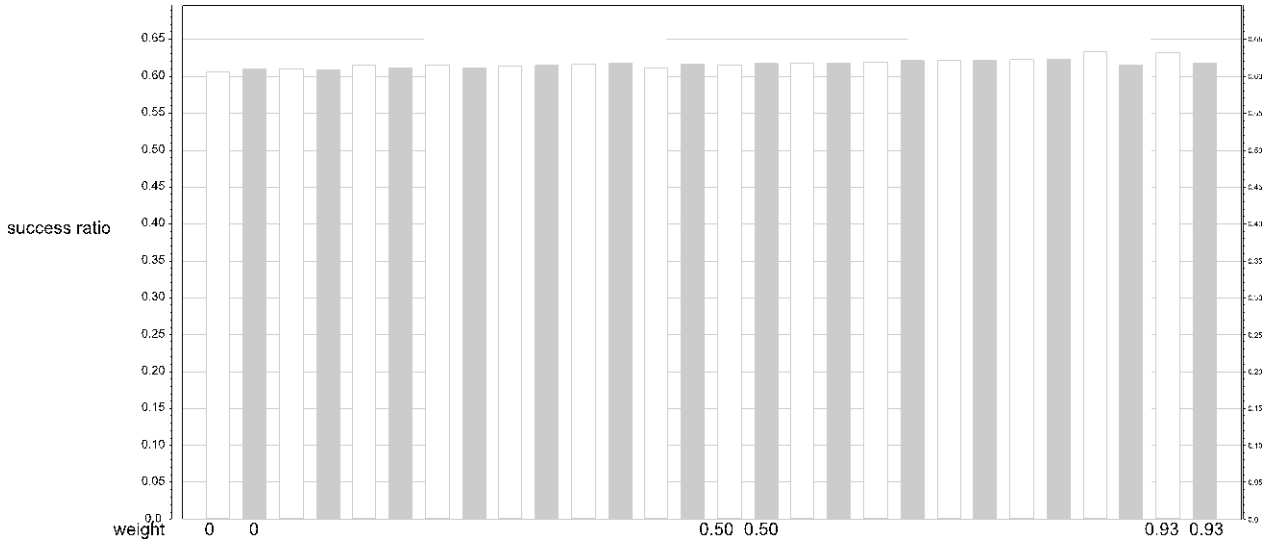

Figure 5: GET Success ratio with modification 1 (white) and 2 (grey)

The bars in Figure 6 show the GET latency of Kademlia in the described environment with modification 1 and 2. The mean latency of a lookup is 20 seconds in the unmodified Kademlia implementation. This value can be read at the two bars labeled with a weight of 0. The bars to the right show a stronger influence of the modification.

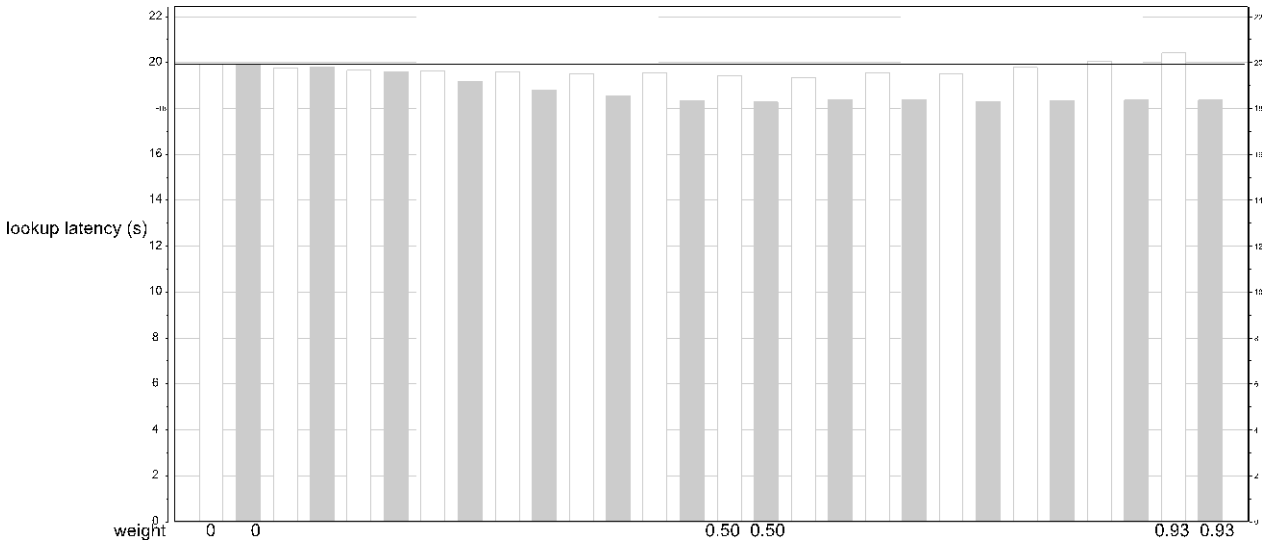

Figure 6: GET latency with modification 1 (white) and 2 (grey)

Modification 1 is shown as white bars, modification 2 as grey bars. Both changes to the original algorithm have an effect on the delay of a lookup. Modification 1 reduces the lookup latency by 0.7 second at a weight of 0.572. The lookup latency increases when a higher weight is used and the latency even increases compared to the unmodified algorithm when the weight comes close to 1. Modification 2 performs better and reduces the lookup latency to 18.3 seconds (-8%) at a weight of 0.5. The lookup latency does not improve further as weight is increased.

The traffic generated by the nodes is shown in Figure 7. Modification 1 has no measurable influence on the traffic caused by the overlay while modification 2 comes with a cost. The traffic caused by a node increases from 100 byte/s to 116 byte/s.

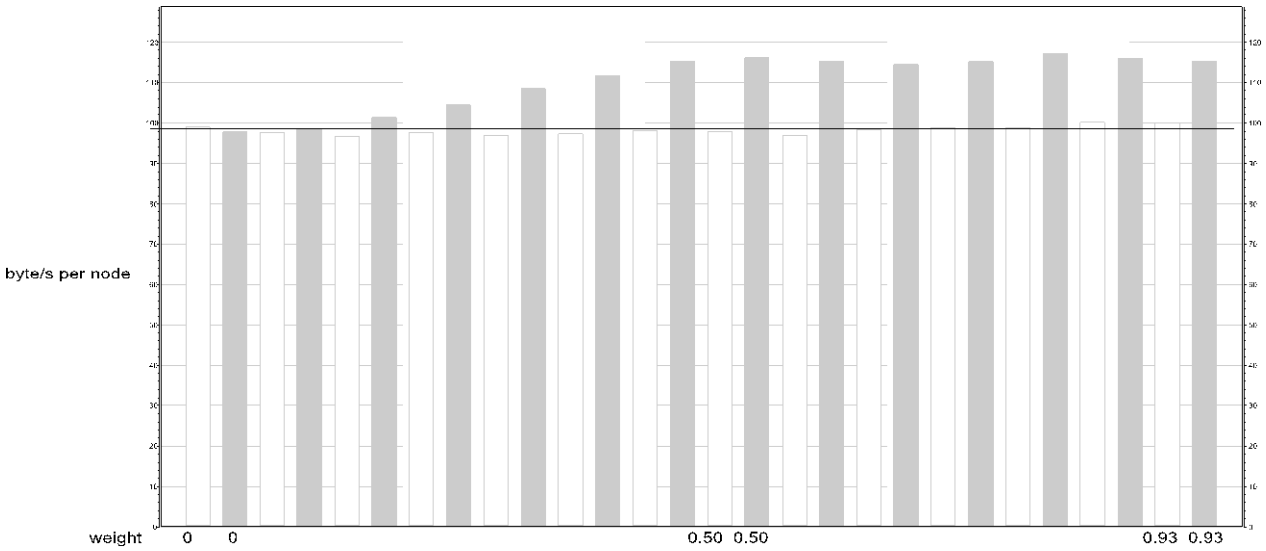

Figure 7: Traffic per node, modification 1 (white) and 2 (grey)

The additional traffic is caused by the changes to the overlay routing. The original Kademlia routing prefers nodes close to the destination node in terms of the XOR metric. This scheme prefers the lowest overlay hop count. Any other scheme will increase the overlay hop count and thereby increase the traffic generated. The effect of modification 1 is so small that it may be argued it has no effect at all. In the existing scenario, modification 1 is not effective. Asking the fast nodes first has the same effect as asking normal nodes first and then - as soon as an answer comes - to proceed in routing. The existing parallelization already prefers fast nodes as the routing algorithm proceeds as soon an answer arrives. Modification 2 addresses a different fact. The routing algorithm tries to hold the number of packets "in flight" constant. So if a packet is lost due to packet errors, it is counted as "in flight" until the timeout (the default is 10s) expires. For this reason a packet loss prevents sending out new packets as long it remains undetected. So preferring nodes with a lower packet error rate pays off in terms of decreased GET latency.

In previous measurements the preference was calculated from the delay or the packet error rate of a connection between the source and destination. We now take a look at the results of modification 3. In this case a node makes a routing decision only based on information reported by the remote node. The preference value is therefore denoted $pref_{(d)}$ instead of $pref_{(s,d)}$. A scenario could involve two different kinds of nodes which differ in the type of power supply. One kind of node is self-powered. The other kind is battery-powered and tries to avoid routing traffic. A battery-powered node publishes a low preference value while the others publish a neutral value. We assume every 10^{th} node to be battery-powered. The accumulated traffic generated by the battery-powered nodes is shown in Figure 8. The grey bars show the traffic sent by the unmodified Kademlia implementation, the black bars depict the results of modification 3. Nodes which arrive earlier (bars more to the left) encounter higher traffic. This is the result of Kademlia's bucket eviction policy and is an expected behaviour not connected to the modification. Long lived nodes are preferred over new nodes as they are supposed to stay longer due to an assumed Poisson or Weibull arrival process. So old node referenced never get replaced by new learned ones if the bucket is already full. Old references get deleted only if a message to that node times out.

The amount of traffic of the battery-powered nodes can be reduced by 25% compared to the unmodified nodes. The accumulated traffic of all nodes over the whole simulation time remains nearly unchanged at 151.11 MB vs. 153.38 MB with modification 3 while the success rate does not differ significantly.

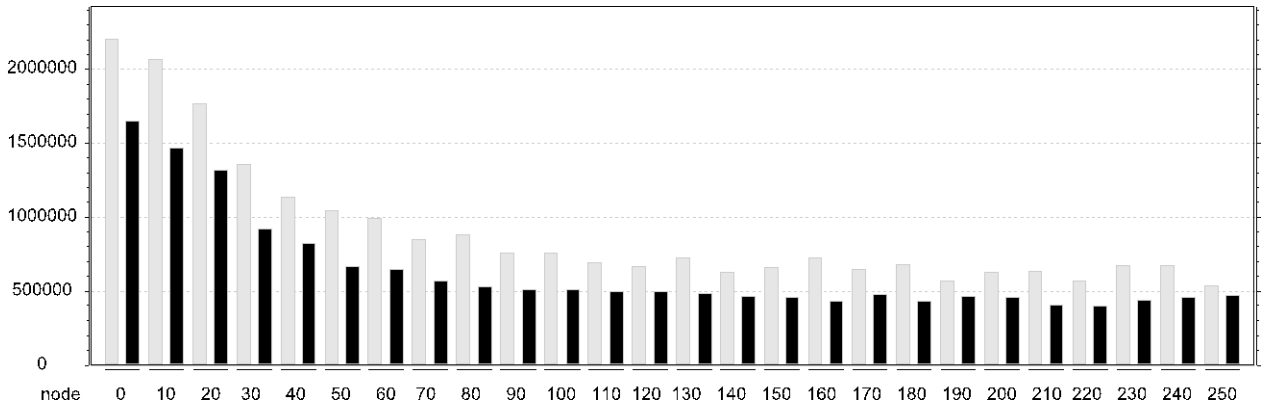

Figure 8: Accumulated traffic per node without (grey) and with (black) modification 3

Our changes to the Kademlia routing decrease the time it takes to retrieve a piece of data from the distributed peer-to-peer storage in a difficult network environment. The reduced delay is a trade-off between delay and the amount of transmission capacity needed to maintain the peer-to-peer overlay. A weight factor is used to optimize the overlay either with respect to delay or bandwidth. The routing modifications can be utilized to avoid excess traffic generation at special nodes, e.g. nodes with limited battery power.

CONCLUSION AND FURTHER STEPS

We described a simulation of a tactical network with peer-to-peer nodes. The nodes in our simulation constantly exchange data and the reliability, speed and load of the overlay network was measured. A change to the routing was made in a way that it does not affect the correctness of the system while offering to include arbitrary cross-layer information into the Kademlia peer-to-peer system. We proposed three modifications or possibilities to use the cross-layer information. The first two modifications increase the speed of the system. One modification reports the delay of channel links as cross-layer information, the other reports the bit error rate of a channel to the overlay. If the bit error rate in included by our mechanism into the overlay routing, we showed that the speed of the system may be increased. This modification also increased node traffic. A smooth trade-off between speed and traffic consumption is realized by a weight factor. As a third possibility we used the change to the Kademlia routing to avoid traffic at nodes which are battery powered. The rest of the system took the load of the weaker nodes and distributed it evenly with only a small increase of the overall load. How much influence the cross-layer information should have mainly relies on the environment the system is used in. Further steps may include the delivery of expected leave time or transmission capacity of a node through the cross-layer interface to the overlay.

REFERENCES

[1] I. Baumgart and S. Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *ICPADS*, pages 1–8. IEEE Computer Society, 2007.

[2] T. Ginzler and M. Amanowicz. Evaluating a Peer-to-Peer System in a Simulated Tactical Environment. In Military Communication Conference, 2009

[3] Ion Stoica and Robert Morris and David R. Karger and M. Frans Kaashoek and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In SIGCOMM, 2001

[4] Baset, S. A. and Schulzrinne, H. G. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In INFOCOM, 2006

[5] S. Guha and N. Daswani. An experimental study of the skype peer-to-peer voIP system. Technical report, Cornell University, Dec. 16 2005.

[6] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In INFOCOM, pages 225–236. IEEE, 2005.

[7] Maymounkov and Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In International Workshop on Peer-to-Peer Systems (IPTPS), LNCS, volume 1, 2002.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, International Computer Science Institute, Berkeley, CA, Oct. 2000.

[9] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In J. Crowcroft and M. Hofmann, editors, Networked Group Communication, Third International COST264 Workshop (NGC'2001), volume 2233 of Lecture Notes in Computer Science, pages 30–43, Nov. 2001.

[10] M. Steiner, E. W. Biersack, and T. En Najjary. Actively monitoring peers in KAD. In IPTPS'07, 6th International Workshop on Peer-to-Peer Systems, February 26-27, 2007, Bellevue, USA, Feb 2007.

[11] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In J. M. Almeida, V. A. F. Almeida, and P. Barford, editors, Internet Measurement Conference, pages 189–202. ACM, 2006.

[12] A. Varga. OMNeT++ discrete event simulation system. http://www.omnetpp.org, April 2009.

[13] Ingmar Baumgart and Bernhard Heep and Stephan Krause. OverSim: A Flexible Overlay Network Simulation Framework, Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA

[14] András Varga. OMNeT++ Discrete Event Simulation System. http://www.omnetpp.org

[15] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001